# The Comparator Interface

The Comparator interface is similar to the Comparable interface, and the two can often be confused with each other.

Its declaration and primary abstract method are shown here, in comparison to Comparable.

You'll notice that the method names are different, compare vs. compareTo.

| Comparator | Comparable |
| --- | --- |
| `public interface Comparator<T> {`<br><br>`    int compare(T o1, T o2);`<br>`}` | `public interface Comparable<T> {`<br><br>`    int compareTo(T o);`<br>`}` |

# The Comparator Interface

The compare method takes two arguments vs. one for compareTo, meaning that it will compare the two arguments to one another, and not one object to the instance itself.

We'll review Comparator in code, but in a slightly manufactured way.

It's common practice to include a Comparator as a nested class.

| Comparator | Comparable |
| --- | --- |
| `public interface Comparator<T> {`<br><br>`    int compare(T o1, T o2);`<br>`}` | `public interface Comparable<T> {`<br><br>`    int compareTo(T o);`<br>`}` |

# Summary of Differences

| Comparable | Comparator |
| --- | --- |
| `int compareTo(T o);` | `int compare(T o1, T o2);` |
| Compares the argument with the current instance.<br><br>Called from the instance of the class that implements Comparable.<br><br>Best practice is to have this.compareTo(o) == 0 result in this.equals(o) being true.<br><br>Arrays.sort(T[] elements) requires T to implement Comparable. | Compares two arguments of the same type with each other.<br><br>Called from an instance of Comparator.<br><br>Does not require the class itself to implement Comparator, though you could also implement it this way.<br><br>Array.sort(T[] elements, Comparator<T>) does not require T to implement Comparable. |